Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	51	(inline or in?line or inlined or in?lined or (in adj line?)) near1 stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 11:59
S2	0	((inline or in?line or inlined or in?lined or (in adj line?)) near1 stub) and (caller or (direct$2 near2 function near2 call$3))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:01
S3	46	((inline or in?line or inlined or in?lined or (in adj line?)) near1 stub) and @ad<="20010628"	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:11
S4	9	(managed near2 (code or function or object or application)) and ((unmanaged or un?managed) near2 (code or function or object))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:15
S5	4	((managed near2 (code or function or object or application)) and ((unmanaged or un?managed) near2 (code or function or object))) and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:23
S6	764	(rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:27
S7	354	((rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller) and (inlined or inline or in-lined or inlined or (in adj line?) or (direct$2 near1 (call$2 or communicat$3)))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:29
S8	320	(((rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller) and (inlined or inline or in-lined or inlined or (in adj line?) or (direct$2 near1 (call$2 or communicat$3)))) not ((external or remote$2) near2 stub)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:44

S9	131	((((rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller) and (inlined or inline or in-lined or inlined or (in adj line?) or (direct$2 near1 (call$2 or communicat$3)))) not ((external or remote$2) near2 stub)) and ((caller or stub) with (return or (call$3 with return with pair)))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:46
S10	0	(((((rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller) and (inlined or inline or in-lined or inlined or (in adj line?) or (direct$2 near1 (call$2 or communicat$3)))) not ((external or remote$2) near2 stub)) and ((caller or stub) with (return or (call$3 with return with pair)))) and ((stack with (marker or pointer or hoist$3)) same loop$3)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 12:47
S11	28	(((((rpc! or (remote adj procedure adj call) or ((function or procedure) near2 call$3 near4 (unmanaged or un?managed or remote))) and caller) and (inlined or inline or in-lined or inlined or (in adj line?) or (direct$2 near1 (call$2 or communicat$3)))) not ((external or remote$2) near2 stub)) and ((caller or stub) with (return or (call$3 with return with pair)))) and (stack with (marker or pointer or hoist$3))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:01
S12	3	caller and (rpc or (remote$2 near2 (procedure or function) near2 call) or (call$3 near2 (unmanaged or un?managed))) and (stack same ((single or sole or one or once) near2 push$3))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:11
S13	50	lrpc or ((lightweight or light?weight) adj remote adj procedure adj call$3).	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:12
S14	49	(lrpc or ((lightweight or light?weight) adj remote adj procedure adj call$3)) and @ad<="200106285"	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:12

S15	22	((lrpc or ((lightweight or light?weight) adj remote adj procedure adj call$3)) and @ad<="200106285") and stack	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:57
S16	1948	(707/206 707/103$2 709/330 717/162 717/165 717/167 717/116).ccls.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:58
S17	104	((707/206 707/103$2 709/330 717/162 717/165 717/167 717/116).ccls.) and stack and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 16:58
S18	47	(((707/206 707/103$2 709/330 717/162 717/165 717/167 717/116).ccls.) and stack and caller) and ((stack or queue or push$3) with (single or one or once or (only adj (one or once))))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 17:00
S19	22	((((707/206 707/103$2 709/330 717/162 717/165 717/167 717/116).ccls.) and stack and caller) and ((stack or queue or push$3) with (single or one or once or (only adj (one or once))))) and synch$9	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 17:00
S20	26	((((707/206 707/103$2 709/330 717/162 717/165 717/167 717/116).ccls.) and stack and caller) and ((stack or queue or push$3) with (single or one or once or (only adj (one or once))))) and (synch$9 or synchronization)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/04 17:01
S21	40	(inline or inlined or in-line or in-lined or (in adj line?)) near12 stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/05 10:50
S22	1	((inline or inlined or in-line or in-lined or (in adj line?)) near12 stub) and stack and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/05 10:50
S23	1	((inline or inlined or in-line or in-lined or (in adj line?)) near12 stub) and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/05 10:50
S24	147	(inline or inlined or in-line or in-lined or (in adj line?)) same stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/05 10:50

S25	4	((inline or inlined or in-line or in-lined or (in adj line?)) same stub) and caller	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2003/09/05 10:50
S26	580	(rpc or lrpc or caller or (invok$3 near2 (code or method))) same stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/02/15 17:22
S27	64	S26 and (stub near3 (inline$3 or in?lin$3 or "in lined" or embed$4 or within or internal or inside))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/02/15 17:26
S28	144	(inlin$4 or in?lin$4 or (in adj lin$4)) near2 stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 14:23
S29	146	(inlin$4 or in?lin$4 or (in adj lin$4) or hard?cod$4 or (hard adj cod$4)) near2 stub	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 14:23
S30	9	S29 and (rpc or (remote adj procedure adj call) or invok$8)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 14:24
S31	9	S29 and (rpc or (remote adj procedure adj call) or invok$8 or caller)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 15:24
S32	115	S29 and @ad<="20010628"	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 15:27
S33	6	S32 and (rpc or "remote procedure call" or object?oriented or "object oriented" or caller or callee or invok$8)	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 15:26
S34	3	S29 and ("707"/$4 "717"/$4 "709"/$4).ccls.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 15:29
S35	19	(rpc or lrpc or "remote procedure call" or caller or invok$8) and (stub with (inlin$4 or in?lin$4 or (in adj lin$4) or hard?cod$4 or (hard adj cod$4) or (early adj bind$4)))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/03 15:53
S36	8821	(717/108 717/165 719/330 707/103$3 707/100 707/101 707/102).ccls.	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/04 10:26

S37	3	S36 and (stub with (inlin$4 or in?lin$4 or (in adj lin$4) or hard?cod$4 or (hard adj cod$4) or embed$4) with (caller or calling or invok$8))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/04 10:29
S38	8	S36 and (stub with (inlin$4 or in?lin$4 or (in adj lin$4) or hard?cod$4 or (hard adj cod$4) or embed$4))	US-PGPUB; USPAT; EPO; JPO; IBM_TDB	OR	ON	2005/08/04 10:30

PORTAL
USPTO

Search: ⊙ The ACM Digital Library   ○ The Guide

+caller +invok* +stub +inlin*

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Published before June 2001
Terms used **caller** **invok** **stub** **inlin**

Found **29** of **115,630**

| Sort results by | relevance |
| Display results | expanded form |

● Save results to a Binder
② Search Tips
▣ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 29   Result page: **1** 2   next

Relevance scale ▢ ▭ ▨ ▩ ▩

1 Distributed systems - programming and management: On remote procedure call
Patrícia Gomes Soares
November 1992 **Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 2**
Full text available: 🗎 pdf(4.52 MB)   Additional Information: full citation, abstract, references, citings

The Remote Procedure Call (RPC) paradigm is reviewed. The concept is described, along with the backbone structure of the mechanisms that support it. An overview of works in supporting these mechanisms is discussed. Extensions to the paradigm that have been proposed to enlarge its suitability, are studied. The main contributions of this paper are a standard view and classification of RPC mechanisms according to different perspectives, and a snapshot of the paradigm in use today and of goals for t ...

2 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming**
Full text available: 🗎 pdf(320.62 KB)   Additional Information: full citation, references, citings, index terms

3 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: 🗎 pdf(190.06 KB)   Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

4 Reconciling responsiveness with performance in pure object-oriented languages
Urs Hölzle, David Ungar

July 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**,
Volume 18 Issue 4

Full text available: pdf(537.19 KB)   Additional Information: full citation, abstract, references, citings, index terms, review

Dynamically dispatched calls often limit the performance of object-oriented programs, since opject-oriented programming encourages factoring code into small, reusable units, thereby increasing the frequency of these expensive operations. Frequent calls not only slow down execution with the dispatch overhead per se, but more importantly they hinder optimization by limiting the range and effectiveness of standard global optimizations. In particular, dynamically dispatched calles prevent stand ...

Keywords: adaptive optimization, pause clustering, profile-based optimization, run-time compilation, type feedback

5 SAFKASI: a security mechanism for language-based systems
Dan S. Wallach, Andrew W. Appel, Edward W. Felten
October 2000 **ACM Transactions on Software Engineering and Methodology (TOSEM)**,
Volume 9 Issue 4

Full text available: pdf(234.89 KB)   Additional Information: full citation, abstract, references, citings, index terms

In order to run untrusted code in the same process as trusted code, there must be a mechanism to allow dangerous calls to determine if their caller is authorized to exercise the privilege of using the dangerous routine. Java systems have adopted a technique called stack inspection to address this concern. But its original definition, in terms of searching stack frames, had an unclear relationship to the actual achievement of security, overconstrained the implementation of a Java system, lim ...

Keywords: Internet, Java, WWW, access control, applets, security-passing style, stack inspection

6 Selective specialization for object-oriented languages
Jeffrey Dean, Craig Chambers, David Grove
June 1995 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation**, Volume 30 Issue 6

Full text available: pdf(1.32 MB)   Additional Information: full citation, abstract, references, citings, index terms

Dynamic dispatching is a major source of run-time overhead in object-oriented languages, due both to the direct cost of method lookup and to the indirect effect of preventing other optimizations. To reduce this overhead, optimizing compilers for object-oriented languages analyze the classes of objects stored in program variables, with the goal of bounding the possible classes of message receivers enough so that the compiler can uniquely determine the target of a message send at compile time ...

7 A client-side stub interpreter
Peter B. Kessler
August 1994 **ACM SIGPLAN Notices , Proceedings of the workshop on Interface definition languages**, Volume 29 Issue 8

Full text available: pdf(554.02 KB)   Additional Information: full citation, abstract, citings, index terms

We have built a research operating system in which all services are presented through interfaces described by an interface description language. The system consists of a micro-kernel that supports a small number of these interfaces, and a large number of interfaces that are implemented by user-level code. A typical service implements one or more

interfaces, but is a client of many other interfaces that are implemented elsewhere in the system. We have an interface compiler that generates client-s ...

8 A study of devirtualization techniques for a Java Just-In-Time compiler

Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Hideaki Komatsu, Toshio Nakatani
October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 35 Issue 10

Full text available: pdf(225.89 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

Many devirtualization techniques have been proposed to reduce the runtime overhead of dynamic method calls for various object-oriented languages, however, most of them are less effective or cannot be applied for Java in a straightforward manner. This is partly because Java is a statically-typed language and thus transforming a dynamic call to a static one does not make a tangible performance gain (owing to the low overhead of accessing the method table) unless it is inlined, and partly because t ...

9 Whole-program optimization for time and space efficient threads

Dirk Grunwald, Richard Neves
September 1996 **Proceedings of the seventh international conference on Architectural support for programming languages and operating systems**, Volume 31 , 30 Issue 9 , 5

Full text available: pdf(1.11 MB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

Modern languages and operating systems often encourage programmers to use *threads,* or independent control streams, to mask the overhead of some operations and simplify program structure. Multitasking operating systems use threads to mask communication latency, either with hardwares devices or users. Client-server applications typically use threads to simplify the complex control-flow that arises when multiple clients are used. Recently, the scientific computing community has started using ...

10 Evaluating the performance limitations of MPMD communication

Chi-Chao Chang, Grzegorz Czajkowski, Thorsten von Eicken, Carl Kesselman
November 1997 **Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)**

Full text available: pdf(125.61 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>

The MPMD approach for parallel computing is attractive for programmers who seek fast development cycles, high code re-use, and modular programming, or whose applications exhibit irregular computation loads and communication patterns. RPC is widely adopted as the communication abstraction for crossing address space boundaries. However, the communication overheads of existing RPC-based systems are usually an order of magnitude higher than those found in highly tuned SPMD systems. This problem has ...

11 Specialization tools and techniques for systematic optimization of system software

Dylan McNamee, Jonathan Walpole, Calton Pu, Crispin Cowan, Charles Krasic, Ashvin Goel, Perry Wagle, Charles Consel, Gilles Muller, Renauld Marlet
May 2001 **ACM Transactions on Computer Systems (TOCS)**, Volume 19 Issue 2

Full text available: pdf(178.52 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

Specialization has been recognized as a powerful technique for optimizing operating systems. However, specialization has not been broadly applied beyond the research community because current techniques based on manual specialization, are time-consuming and error-prone. The goal of the work described in this paper is to help

operating system tuners perform specialization more easily. We have built a specialization
toolkit that assists the major tasks of specializing operating systems. We de ...

Keywords: operating system specialization, optimization, software architecture

12 <u>The design and performance of a scable ORB architecture for COBRA asynchronous</u>
<u>messaging</u>
Alexander B. Arulanthu, Carlos O'Ryan, Douglas C. Schmidt, Michael Kircher, Jeff Parsons
April 2000 **IFIP/ACM International Conference on Distributed systems platforms**

Full text available: pdf(174.72 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>

Historically, method-oriented middleware, such as Sun RPC, DCE, Java RMI, COM, and
CORBA, has provided synchronous method invocation (SMI) models to applications.
Although SMI works well for conventional client/server applications, it is not well-suited for
high-performance or real-time applications due to its lack of scalability. To address this
problem, the OMG has recently standardized an asynchronous method invocation (AMI)
model for CORBA. AMI provides CORBA with many of the capabilit ...

13 <u>Measuring the performance of communication middleware on high-speed networks</u>
Aniruddha Gokhale, Douglas C. Schmidt
August 1996 **ACM SIGCOMM Computer Communication Review , Conference**
proceedings on Applications, technologies, architectures, and protocols
for computer communications, Volume 26 Issue 4

Full text available: pdf(270.13 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index</u>
<u>terms</u>

Conventional implementations of communication middleware (such as CORBA and
traditional RPC toolkits) incur considerable over-head when used for performance-sensitive
applications over high-speed networks. As gigabit networks become pervasive, inefficient
middleware will force programmers to use lower-level mechanisms to achieve the necessary
transfer rates. This is a serious problem for mission/life-critical applications (such as
satellite surveillance and medical imaging).This paper compares t ...

14 <u>Analysis of techniques to improve protocol processing latency</u>
David Mosberger, Larry L. Peterson, Patrick G. Bridges, Sean O'Malley
August 1996 **ACM SIGCOMM Computer Communication Review , Conference**
proceedings on Applications, technologies, architectures, and protocols
for computer communications, Volume 26 Issue 4

Full text available: pdf(134.36 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index</u>
<u>terms</u>

This paper describes several techniques designed to improve protocol latency, and reports
on their effectiveness when measured on a modern RISC machine employing the DEC Alpha
processor. We found that the memory system---which has long been known to dominate
network throughput---is also a key factor in protocol latency. As a result, improving
instruction cache effectiveness can greatly reduce protocol processing overheads. An
important metric in this context is the *memory cycles per instructi ...*

15 <u>Dynamo: a transparent dynamic optimization system</u>
Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference**
on Programming language design and implementation, Volume 35 Issue 5

Full text available: pdf(156.03 KB)   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index</u>
<u>terms</u>

We describe the design and implementation of Dynamo, a software dynamic optimization

system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream to Dynamo can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in the latter, more challenging situation, in order to emphasize the ...

16 Design, implementation, and evaluation of optimizations in a just-in-time compiler
Kazuaki Ishizaki, Motohiro Kawahito, Toshiaki Yasue, Mikio Takeuchi, Takeshi Ogasawara, Toshio Suganuma, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
June 1999 **Proceedings of the ACM 1999 conference on Java Grande**

Full text available: pdf(1.09 MB)   Additional Information: full citation, references, citings, index terms

17 SCONE: using concurrent objects for low-level operating system programming
Jun-ichiro Itoh, Yasuhiko Yokote, Mario Tokoro
October 1995 **ACM SIGPLAN Notices , Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications,**
Volume 30 Issue 10

Full text available: pdf(1.66 MB)   Additional Information: full citation, abstract, references, citings, index terms

This paper proposes a methodology for making low-level system code of operating systems be replaceable at runtime. Our approach is to use concurrent objects as a basic programming unit for low-level system programs. To realize the different need for each type of system code and to execute these concurrent objects sufficiently efficient, we use a combination of dedicated system service layers and other implementation techniques. System service layers provide the most suitable primitive operations ...

18 Introducing Ada 9X
John Barnes
November 1993 **ACM SIGAda Ada Letters**, Volume XIII Issue 6

Full text available: pdf(4.39 MB)   Additional Information: full citation, citings, index terms

19 PIROL: a case study for multidimensional separation of concerns in software engineering environments
Stephan Herrmann, Mira Mezini
October 2000 **ACM SIGPLAN Notices , Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 35 Issue 10

Full text available: pdf(441.79 KB)   Additional Information: full citation, abstract, references, citings, index terms

In this paper, we present our experience with applying multidimensional separation of concerns to a software engineering environment. By comparing two different designs of our system, we show the importance of separating integration issues from the implementation of the individual concerns. We present a model in which integration issues are encapsulated into rst--class connector objects and indicate how this facilitates the understandability, maintenance and evolution of the system. We identify ...

Keywords: component integration, domain—specific language, separation of concerns, software engineering environment

20

Annotation-directed run-time specialization in C

Brian Grant, Markus Mock, Matthai Philipose, Craig Chambers, Susan J. Eggers

December 1997 **ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation**, Volume 32 Issue 12

Full text available: pdf(1.99 MB)   Additional Information: full citation, abstract, references, citings, index terms

We present the design of a dynamic compilation system for C. Directed by a few declarative user annotations specifying where and on what dynamic compilation is to take place, a binding time analysis computes the set of run-time constants at each program point in each annotated procedure's control flow graph; the analysis supports program-point-specific polyvariant division and specialization. The analysis results guide the construction of a specialized run-time specializer for each dynamically c ...

Results 1 - 20 of 29   Result page: **1**   2   next

PORTAL

USPTO

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Published before June 2001
Terms used **rpc stub inlin in line in line**

Found **289** of **115,630**

Sort results by	relevance		Save results to a Binder	Try an Advanced Search
Display results	expanded form		Search Tips	Try this search in The ACM Guide
			Open results in a new window	

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9 10   next
Best 200 shown   Relevance scale ☐ ☐ ▦ ▦ ▦

1 Scaling up partial evaluation for optimizing the Sun commercial RPC protocol
Gilles Muller, Eugen-Nicolae Volanschi, Renaud Marlet
December 1997 **ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation,** Volume 32 Issue 12

Full text available: 📄 pdf(1.05 MB)   Additional Information: full citation, abstract, references, citings, index terms

We report here a successful experiment in using partial evaluation on a realistic program, namely the Sun commercial RPC (Remote Procedure Call) protocol. The Sun RPC is implemented in a highly generic way that offers multiple opportunities of specialization.Our study also shows the incapacity of traditional binding-time analyses to treat real system programs. Our experiment has been made with Tempo, a partial evaluator for C programs targeted towards system software. Tempo's binding-time analys ...

2 Specialization tools and techniques for systematic optimization of system software
Dylan McNamee, Jonathan Walpole, Calton Pu, Crispin Cowan, Charles Krasic, Ashvin Goel, Perry Wagle, Charles Consel, Gilles Muller, Renauld Marlet
May 2001 **ACM Transactions on Computer Systems (TOCS),** Volume 19 Issue 2

Full text available: 📄 pdf(178.52 KB)   Additional Information: full citation, abstract, references, citings, index terms

Specialization has been recognized as a powerful technique for optimizing operating systems. However, specialization has not been broadly applied beyond the research community because current techniques based on manual specialization, are time-consuming and error-prone. The goal of the work described in this paper is to help operating system tuners perform specialization more easily. We have built a specialization toolkit that assists the major tasks of specializing operating systems. We de ...

Keywords: operating system specialization, optimization, software architecture

3 Flick: a flexible, optimizing IDL compiler
Eric Eide, Kevin Frei, Bryan Ford, Jay Lepreau, Gary Lindstrom
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conference on Programming language design and implementation,** Volume 32 Issue 5

Full text available: 📄 pdf(1.75 MB)   Additional Information: full citation, abstract, references, citings, index

terms

An interface definition language (IDL) is a nontraditional language for describing interfaces between software components. IDL compilers generate "stubs" that provide separate communicating processes with the abstraction of local object invocation or procedure call. High-quality stub generation is essential for applications to benefit from component-based designs, whether the components reside on a single computer or on multiple networked hosts. Typical IDL compilers, ...

4 Analysis of techniques to improve protocol processing latency

David Mosberger, Larry L. Peterson, Patrick G. Bridges, Sean O'Malley

August 1996 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Applications, technologies, architectures, and protocols for computer communications**, Volume 26 Issue 4

Full text available: pdf(134.36 KB)   Additional Information: full citation, abstract, references, citings, index terms

This paper describes several techniques designed to improve protocol latency, and reports on their effectiveness when measured on a modern RISC machine employing the DEC Alpha processor. We found that the memory system---which has long been known to dominate network throughput---is also a key factor in protocol latency. As a result, improving instruction cache effectiveness can greatly reduce protocol processing overheads. An important metric in this context is the *memory cycles per instructi ...*

5 Evaluating the performance limitations of MPMD communication

Chi-Chao Chang, Grzegorz Czajkowski, Thorsten von Eicken, Carl Kesselman

November 1997 **Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)**

Full text available: pdf(125.61 KB)   Additional Information: full citation, abstract, references

The MPMD approach for parallel computing is attractive for programmers who seek fast development cycles, high code re-use, and modular programming, or whose applications exhibit irregular computation loads and communication patterns. RPC is widely adopted as the communication abstraction for crossing address space boundaries. However, the communication overheads of existing RPC-based systems are usually an order of magnitude higher than those found in highly tuned SPMD systems. This problem has ...

6 USC: a universal stub compiler

Sean O'Malley, Todd Proebsting, Allen Brady Montz

October 1994 **ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Communications architectures, protocols and applications,** Volume 24 Issue 4

Full text available: pdf(1.23 MB)   Additional Information: full citation, abstract, references, citings, index terms

USC is a new stub compiler that generates stubs that perform many data conversion operations. USC is flexible and can be used in situations where previously only manual code generation was possible. USC generated code is up to 20 times faster than code generated by traditional argument marshaling schemes such as ASN.1 and Sun XDR. This paper presents the design of USC and a comprehensive set of experiments that compares USC performance with the best manually generated code and traditional s ...

7 Practical use of a polymorphic applicative language

Butler W. Lampson, Eric E. Schmidt

January 1983 **Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages**

Full text available: pdf(1.84 MB)   Additional Information: full citation, abstract, references, citings

Assembling a large system from its component elements is not a simple task. An adequate notation for specifying this task must reflect the system structure, accommodate many configurations of the system and many versions as it develops, and be a suitable input to the many tools that support software development. The language described here applies the ideas of λ-abstraction, hierarchical naming and type-checking to this problem. Some preliminary experience with its use is also given.

8 <u>Eraser: a dynamic data race detector for multi-threaded programs</u>

Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, Thomas Anderson
October 1997 **ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles**, Volume 31 Issue 5

Full text available: <u>pdf(1.51 MB)</u>   Additional Information: <u>full citation</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

9 <u>Connecting software components with declarative glue</u>

Brian W. Beach
June 1992 **Proceedings of the 14th international conference on Software engineering**

Full text available: <u>pdf(1.33 MB)</u>   Additional Information: <u>full citation</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

10 <u>Eraser: a dynamic data race detector for multithreaded programs</u>

Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, Thomas Anderson
November 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 4

Full text available: <u>pdf(136.04 KB)</u>   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake during debugging. This article describes a new tool, called Eraser, for dynamically detecting data races in lock-based multithreaded programs. Eraser uses binary rewriting techniques to monitor every shared-monory reference and verify that consistent locking behavior is observed. We present several case studies ...

Keywords: binary code modification, multithreaded programming, race detection

11 <u>Chores: enhanced run-time support for shared-memory parallel computing</u>

Derek L. Eager, John Jahorjan
February 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 1

Full text available: <u>pdf(2.24 MB)</u>   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>references</u>, <u>citings</u>, <u>index terms</u>, <u>review</u>

Parallel computing is increasingly important in the solution of large-scale numerical problems. The difficulty of efficiently hand-coding parallelism, and the limitations of parallelizing compilers, have nonetheless restricted its use by scientific programmers. In this paper we propose a new paradigm, chores, for the run-time support of parallel computing on shared-memory multiprocessors. We consider specifically uniform memory access shared-memory environments, ...

12 <u>Matchmaker: an interface specification language for distributed processing</u>

Michael B. Jones, Richard F. Rashid, Mary R. Thompson
January 1985 **Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages**

Full text available: <u>pdf(923.04 KB)</u>   Additional Information: <u>full citation</u>, <u>abstract</u>, <u>citings</u>, <u>index terms</u>

Matchmaker, a language used to specify and automate the generation of interprocess communication interfaces, is presented. The process of and reasons for the evolution of Matchmaker are described. Performance and usage statistics are presented. Comparisons are made between Matchmaker and other related systems. Possible future directions are examined.

Keywords: distributed systems, interface specification language, interprocess communication, multi-targeted compiler, object-oriented languages, remote procedure call

13 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(4.21 MB)   Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

14 A practical tool for distributing Ada programs: TeleSoft's distributed Ada configuration tool

Tom Burger, Jim Bladen, Richard A. Volz, Ron Theriault, Gary Smith, Kali H. Buhariwalla
December 1992 **Proceedings of the conference on TRI-Ada '92**

Full text available: pdf(859.11 KB)   Additional Information: full citation, references, citings, index terms

15 Signaling and operating system support for native-mode ATM applications

R. Sharma, S. Keshav
October 1994 **ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Communications architectures, protocols and applications,** Volume 24 Issue 4

Full text available: pdf(1.05 MB)   Additional Information: full citation, abstract, references, citings, index terms

Applications communicating over connectionless networks, such as IP, cannot obtain per-connection Quality of Service (QoS) guarantees. In contrast, the connection-oriented nature of the ATM layer and its per-virtual-circuit QoS guarantees are visible to a native-mode ATM application. We describe the design and implementation of operating system and signaling support for native-mode applications, independent of the semantics of the protocol layers or of the signaling protoco ...

16 Client-server computing in mobile environments

Jin Jing, Abdelsalam Sumi Helal, Ahmed Elmagarmid
June 1999 **ACM Computing Surveys (CSUR)**, Volume 31 Issue 2

Full text available: pdf(233.31 KB)   Additional Information: full citation, abstract, references, citings, index terms, review

Recent advances in wireless data networking and portable information appliances have engendered a new paradigm of computing, called mobile computing, in which users carrying portable devices have access to data and information services regardless of their physical location or movement behavior. In the meantime, research addressing information access in mobile environments has proliferated. In this survey, we provide a concrete framework

and categorization of the various way ...

Keywords: application adaptation, cache invalidation, caching, client/server, data dissemination, disconnected operation, mobile applications, mobile client/server, mobile compuing, mobile data, mobility awareness, survey, system application

17 The Alpine file system
M. R. Brown, K. N. Kolling, E. A. Taft
November 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 4

Full text available: pdf(2.95 MB)   Additional Information: full citation, abstract, references, citings, index terms, review

Alpine is a file system that supports atomic transactions and is designed to operate as a service on a computer network. Alpine's primary purpose is to store files that represent databases. An important secondary goal is to store ordinary files representing documents, program modules, and the like. Unlike other file servers described in the literature, Alpine uses a log-based technique to implement atomic file update. Another unusual aspect of Alpine is that it performs all commu ...

18 Network objects
Andrew Birrell, Greg Nelson, Susan Owicki, Edward Wobber
December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles**, Volume 27 Issue 5

Full text available: pdf(1.35 MB)   Additional Information: full citation, abstract, references, citings, index terms

A network object is an object whose methods can be invoked over a network. This paper describes the design, implementation, and early experience with a network objects system for Modula-3. The system is novel for its overall simplicity. The paper includes a thorough description of realistic marshaling algorithms for network objects.

19 Performance of the Firefly RPC
Michael D. Schroeder, Michael Burrows
February 1990 **ACM Transactions on Computer Systems (TOCS)**, Volume 8 Issue 1

Full text available: pdf(1.35 MB)   Additional Information: full citation, abstract, references, citings, index terms, review

In this paper we report on the performance of the remote procedure call (RPC) implementation for the Firefly multiprocessor and analyze the implementation to account precisely for all measured latency. From the analysis and measurements, we estimate how much faster RPC could be if certain improvements were made. The elapsed time for an intermachine call to a remote procedure that accepts no arguments and produces no results is 2.66 ms. The elapsed time for an RPC that has a single 1440-byte ...

20 Performance of Firefly RPC
M. Schroeder, M. Burrows
November 1989 **ACM SIGOPS Operating Systems Review , Proceedings of the twelfth ACM symposium on Operating systems principles**, Volume 23 Issue 5

Full text available: pdf(1.03 MB)   Additional Information: full citation, abstract, references, citings, index terms

In this paper, we report on the performance of the remote procedure call implementation for the Firefly multiprocessor and analyze the implementation to account precisely for all measured latency. From the analysis and measurements, we estimate how much faster RPC could be if certain improvements were made. The elapsed time for an inter-machine call to

a remote procedure that accepts no arguments and produces no results is 2.66 milliseconds. The elapsed time for an RPC that has a ...

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9 10   next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
Terms of Usage   Privacy Policy   Code of Ethics   Contact Us

Useful downloads: Adobe Acrobat   QuickTime   Windows Media Player   Real Player